JAVA DOCUMENTATION COMMENTS

Java supports three types of comments. The first two are the // and the /* */. The third type is called a documentation comment. It begins with the character sequence /** and it ends with */.

Documentation comments allow you to embed information about your program into the program itself. You can then use the javadoc utility program to extract the information and put it into an HTML file.

Documentation comments make it convenient to document your programs.

The javadoc Tags:

The javadoc utility recognizes the following tags:

Tag	Description	Example
@author	Identifies the author of a class.	@author description
@deprecated	Specifies that a class or member is deprecated.	@deprecated description
{@docRoot}	Specifies the path to the root directory of the current documentation	Directory Path
@exception	Identifies an exception thrown by a method.	@exception exception-name explanation
{@inheritDoc}	Inherits a comment from the immediate superclass.	Inherits a comment from the immediate surperclass.
{@link}	Inserts an in-line link to another topic.	{@link name text}
{@linkplain}	Inserts an in-line link to another topic, but the link is displayed in a plain-text font.	Inserts an in-line link to another topic.
@param	Documents a method's parameter.	@param parameter-name explanation
@return	Documents a method's return value.	@return explanation
@see	Specifies a link to another topic.	@see anchor
@serial	Documents a default serializable field.	@serial description
@serialData	Documents the data written by the writeObject( ) or writeExternal( ) methods	@serialData description
@serialField	Documents an ObjectStreamField component.	@serialField name type description
@since	States the release when a specific change was introduced.	@since release
@throws	Same as @exception.	The @throws tag has the same meaning as the @exception tag.
{@value}	Displays the value of a constant, which must be a	Displays the value of a constant, which

	static field.	must be a static field.
@version	Specifies the version of a class.	@version info

Documentation Comment:

After the beginning /**, the first line or lines become the main description of your class, variable, or method.

After that, you can include one or more of the various @ tags. Each @ tag must start at the beginning of a new line or follow an asterisk (*) that is at the start of a line.

Multiple tags of the same type should be grouped together. For example, if you have three @see tags, put them one after the other.

Here is an example of a documentation comment for a class:

```
/**
* This class draws a bar chart.
* @author Zara Ali
* @version 1.2
*/
```

What javadoc Outputs?

The javadoc program takes as input your Java program's source file and outputs several HTML files that contain the program's documentation.

Information about each class will be in its own HTML file. Java utility **javadoc** will also output an index and a hierarchy tree. Other HTML files can be generated.

Since different implementations of javadoc may work differently, you will need to check the instructions that accompany your Java development system for details specific to your version.

Example:

Following is a sample program that uses documentation comments. Notice the way each comment immediately precedes the item that it describes.

After being processed by javadoc, the documentation about the SquareNum class will be found in SquareNum.html.

```
import java.io.*;

/**
* This class demonstrates documentation comments.
* @author Ayan Amhed
* @version 1.2
*/
public class SquareNum {
   /**
   * This method returns the square of num.
   * This is a multiline description. You can use
   * as many lines as you like.
   * @param num The value to be squared.
   * @return num squared.
   */
   public double square(double num) {
      return num * num;
   }
   /**
   * This method inputs a number from the user.
   * @return The value input as a double.
```

```
   * @exception IOException On input error.
   * @see IOException
   */
   public double getNumber() throws IOException {
      InputStreamReader isr = new InputStreamReader(System.in);
      BufferedReader inData = new BufferedReader(isr);
      String str;
      str = inData.readLine();
      return (new Double(str)).doubleValue();
   }
   /**
   * This method demonstrates square().
   * @param args Unused.
   * @return Nothing.
   * @exception IOException On input error.
   * @see IOException
   */
   public static void main(String args[]) throws IOException
   {
      SquareNum ob = new SquareNum();
      double val;
      System.out.println("Enter value to be squared: ");
      val = ob.getNumber();
      val = ob.square(val);
      System.out.println("Squared value is " + val);
   }
}
```

Now process above SquareNum.java file using javadoc utility as follows:

```
$ javadoc SquareNum.java
Loading source file SquareNum.java...
Constructing Javadoc information...
Standard Doclet version 1.5.0_13
Building tree for all the packages and classes...
Generating SquareNum.html...
SquareNum.java:39: warning - @return tag cannot be used\
                      in method with void return type.
Generating package-frame.html...
Generating package-summary.html...
Generating package-tree.html...
Generating constant-values.html...
Building index for all the packages and classes...
Generating overview-tree.html...
Generating index-all.html...
Generating deprecated-list.html...
Building index for all classes...
Generating allclasses-frame.html...
Generating allclasses-noframe.html...
Generating index.html...
Generating help-doc.html...
Generating stylesheet.css...
1 warning
$
```

You can check all the generated documentation here: SquareNum.